UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

THE PERIODIC LOADING PROBLEM: FORMULATION AND HEURISTICS

by

Paul J. Schweitzer[1], Moshe Dror,[2]
and
Pierre Trudeau[3]

[1] William E. Simon Graduate School of Business Administration, University of Rochester, Rochester, N.Y.

[2] Industrial Engineering and Management, Ben Gurion University of the Negev, Beersheva and INRS-Telecommunications, Université du Québec, Verdun, Québec.

[3] Centre de recherche sur les transports, Université de Montréal, Québec.

This paper is subject to appear in INFOR.

Cette étude a été publiée grâce à une subvention du fonds F.C.A.R. pour la l'aide et le soutien à recherche.

Université de Montréal
Centre de recherche sur les transports - Publication #561
February 1988

Dépôt légal - 1er trimestre 1988

ABSTRACT

The periodic loading problem involves the allocation of periodically-occurring tasks to the time-slots of a synchronous server in such a way as to minimize utilization of any time-slot. It arises when a time-division-multiplex communications bus collects data from a collection of periodically-producing sensors, e.g., the MIL-STD-1553 protocol. This paper formulates the problem, gives bounds on the solution, and shows it contains the minimum makespan problem, hence is NP-complete. Heuristic procedures are given and their performance illustrated on a (actual) problem for which a solution is found within 1% accuracy. In addition, an empirical study for the heuristic procedures is conducted on an extensive set of artificial problems.

Key words : synchronous server, periodic loading, complexity, bounds and heuristics.

RÉSUMÉ

Nous étudions la répartition, aux diverses périodes d'un serveur synchrone, de tâches survenant périodiquement. Le but est de minimiser l'utilisation de la période la plus occupée. Un tel problème survient lorsqu'un bus de communication multiplex recueille de l'information d'un ensemble de senseurs émettant à intervalles réguliers, par exemple le protocole MIL-STD-1553. Dans cet article on retrouve une formulation du problème et le calcul de bornes sur la solution, et on montre que le problème est NP-complet. Diverses méthodes heuristiques sont proposées et appliquées à un problème réel, dont une solution à moins d'un pourcent de la solution optimale est trouvée. Enfin, une étude empirique compare les heuristiques sur un ensemble de problèmes simulés.

Mots-clés : serveur synchrone, répartition de tâches périodiques, complexité du problème, bornes et heuristiques.

1. FORMULATION OF THE PERIODIC LOADING PROBLEM (PLP)

Problem Definition

Consider a server whose time is divided into consecutive equal-sized intervals labeled as frame 1, frame 2, ... A set I of customers receives service from this server. Customer i, $1 \leq i \leq |I|$, needs service exactly once every p(i) frames, and each such service has a service duration $d(i) > 0$. The positive integer p(i) is called the *period* for customer i. The decision problem is:

Instance: Finite set I. for $i \in I$ pair of numbers: p(i), d(i); p(i) - integer (denotes the period for i) and d(i) - positive real (denotes the duration for i). Set p = least common multiple of p(1), p(2), ..., p($|I|$) and a set of time periods 1,2, ..., p each of size T.

Decision Problem: Is there a periodic assignment of the set I to the time periods, such that the maximal use of each time period is not greater than T ?

Though the PLP is defined as a decision problem, we confine the analysis to the optimization version (equivalent) of this problem, namely to minimize the maximum usage of any frame.

Denote for each $i \in I$ the first time period to which i is assigned as f(i); f(i) $\in \{1, 2, ..., p(i)\}$. We call f(i) the *phase* for customer i.

The optimization problem addressed in this paper is *phase selection in order to minimize the maximum workload assigned to any frame.* Due to the periodicity in the workload, it suffices to consider the maximum workload among frames 1 through p.

Mathematical Formulation

The above problem can be formulated as the mixed integer program

(1-3a) $\quad z = \min y$

subject to:

(1-3b) $\quad \sum_{j=1}^{p(i)} x_{ij} = 1, \qquad 1 \leq i \leq |I|,$

(1-3c) $\quad y \geq \sum_{i \in I} \sum_{j=1}^{p(i)} d(i) x_{ij} a_{ijk}, \qquad 1 \leq k \leq p,$

(1-3d) $\quad x_{ij} = 0 \text{ or } 1, \qquad 1 \leq i \leq |I|, 1 \leq j \leq p(i)$

where

$$a_{ijk} = \begin{cases} 1 \text{ if } (k-j)_{Mod\, p(i)} = 0 \\ 0 \text{ otherwise} \end{cases}$$

The decision variables here are

(1-4) $$x_{ij} = \begin{cases} 1 \text{ if customer i is assigned to phase j (i.e., f(i) = j)} \\ 0 \text{ otherwise} \end{cases}$$

and a_{ijk} is 1 if and only if assigning customer i to phase j implies customer i receives service in frame k. We will need the properties

(1-5) $$\sum_{k=1}^{p} a_{ijk} = \frac{p}{p(i)}, \qquad 1 \le i \le |I|, 1 \le j \le p(i)$$

(1-6) $$\sum_{j=1}^{p(i)} a_{ijk} = 1, \qquad 1 \le i \le |I|, 1 \le k \le p$$

Constraints (1-3b, 1-3d) assure that to each customer i, exactly one phase is assigned. Constraint (1-3c) assures that y must be at least as large as the total load occurring in frame k. Then z, the minimum of y, will be the minimum value of the maximum workload per frame. Also, with T denoting the duration of one frame, z/T will measure the *maximum utilization of any frame*, and of course must not exceed unity. Other performance measures besides utilization, e.g., waiting times for each customer, are not addressed in this model.

Variations of the problem defined here as PLP were addressed by other authors. In [12] a partition procedure is presented for the general PLP. This procedure reduces the general problem into a set of smaller independent (also NP-complete) problems of a similar structure. No algorithmic procedure for solving the new independent PLP problems is presented. In [7] the authors describe a similar problem but consider slotted periods rather than general schedules. In [5] a very similar formulation to (1-3) is presented and it is noted that the problem is NP-complete. Though an exact algorithm of a branch-and-bound type is described, it is not followed by computational testing.

Occurrence in Synchronous Systems

The PLP occurs as the *slot assignment* problem for a time-division multiplexed (TDM) communication system handling a set of periodic message sources, e.g., a synchronous bus transmitting periodic sensor data. In particular, it arises when constructing polling sequences for avionics equipment connected to a communications bus employing the MIL-STD-1533 protocol [15]. In this application, the frame duration T is set equal to the time between occurrences of the most frequently occurring messages, and such messages will then have periods of unity ($T >> max_i d(i), i \in I$). The goal of minimizing maximum frame utilization is reasonable because the idle time in each frame is available for non-periodic messages, and this goal reduces transmission delays for such messages.

The terms *message* and *customer* are used interchangably in this paper.

The references at the end of this paper describe other occurrences of PLP besides TDM communication buses: distribution system capacity in [6], multi-processor sys-

tems in [7], periodic maintenance in [12].

2. SPECIAL CASES

Introduction

This section describes some special cases of the PLP: the set partitioning problem; the minimum makespan scheduling problem; and the power-period PLP. In particular, the PLP is NP-complete because it is in NP and the reduction of the set partitioning problem and the minimum makespan problem, which are NP-complete ([6]), to PLP.

Reduction of the Set Partitioning Problem to the PLP

Consider the special case of the PLP where p(i) = 2 for every i. The PLP then entails assigning customers to odd or even frames to minimize

$$\max\begin{cases}\sum\{d(i)|\ \text{i is assigned to odd frames}\}\\ \\ \sum\{d(i)|\ \text{i is assigned to even frames}\}\end{cases}$$

This can be shown to be equivalent to minimizing the absolute difference in load between the odd and even frames, which is essentially the set partitioning problem, and is NP-complete [6].

Reduction of the Minimum Makespan Problem to the PLP

Consider the special case of the PLP where p(i) = p for all i. Then (1-3) has the following form:

$$\min\ y$$

Subject to:

$$\sum_{j=1}^{p} x_{ij} = 1, \qquad 1 \le i \le |I|,$$

$$y \ge \sum_{i \in I} d(i) x_{ik}, \qquad 1 \le k \le p,$$

$$x_{ij} = 0 \text{ or } 1, \qquad 1 \le i \le |I|, 1 \le j \le p,$$

and may be interpreted as assigning $|I|$ jobs to p identical parallel processors (with d(i) = duration of job i and $x_{ij} = 1$ if and only if job i is assigned to processor j) in such a way as to minimize the makespan, the time to process all the jobs ([1],[6]). The lemmas which follow, which hold even for nonidentical machines, may be useful for such problems.

Remark: Since the Set Partitioning Problem is a special case of the Minimum Makespan Problem, the first reduction is presented mainly on pedagogical grounds. A very similar reduction was used independently in [7] to prove that a very restricted version of the economic lot sizing problem (ELSP) is NP-complete.

The Power-Period PLP

The power-period PLP is a special case of the PLP where all periods are powers of some integer, r, i.e., $p(i) \in \{1, r, r^2, r^3, ...\}$ for all $i = 1, 2, ..., |I|$. Note that $p = max_{i \in I} p(i)$ in this case. The sequencing rule (polling list within each frame) is especially convenient in this case because the jobs within a given frame are *sequenced in order of nondecreasing period.* Then each customer has, for every frame in which it appears, the *same relative position* within the frame, the *same sequence of predecessors* in the frame, and *receives service at precisely equal time intervals.*

Example 1: The first two properties are illustrated for the case of seven customers labeled I = {A,B,C,D,E,F,G} with periods {1,2,2,2,4,4,8} respectively. Here p = 8 and r = 2. The phase assignments {f(1), f(2), ..., f(7)} = {1, 1, 2, 1, 2, 3, 4} then lead to the following loads on each frame:

Frame 1: A B D	Frame 5: A B D
2: A C E	6: A C E
3: A B D F	7: A B D F
4: A C G	8: A C

The first two properties for customer D say that D is always in third position in its frames, and always has A B as its predecessors in such frames. (Note, however, that D may have different *successors*, as in frames 1 and 3).

Any PLP can be replaced by a power-period PLP, at the expense of serving a customer more often than absolutely necessary, by replacing each period p(i) by $r^{m(i)}$ where m(i) is the largest integer m such that $r^m \leq p(i)$. This has the advantage of smaller overall period p and of greater ease in constructing and executing the order of service. All published implementations of MIL-STD-1553 communications protocol known to the authors use power-period PLPs with r = 2.

In addition, power-periods, especially with r = 2, are a very common strategy for scheduling cyclic production of $|I|$ distinct items (see [4], [10]): a batch of item i is produced at time intervals $Tr^{m(i)}$. Here T is the "frame duration" and p(i) = $r^{m(i)}$ with $m(i) \geq 0$ is the "period". Note, however, that cyclic production scheduling differs in two respects from the PLP: one may choose the integer m(i) as well as the phase; the service time d(i) to produce one batch of item i grows linearly with the time $Tr^{m(i)}$ between batches, assuming constant demand rate and negligible setups. Moreover, a comparison between the PLP and the economic lot sizing problem (ELSP)

is irrelevant. In the ELSP the data is unrestricted but the search for a solution is restricted to one satisfying the constraint that no shortages occur. In the PLP the data is clearly restricted by its periodicity.

3. BOUNDS ON z

Lemma 1 below gives useful bounds on z, the optimal solution to (1-3). We first need the following notation. The average workload duration per frame is denoted by

(3-1) $\quad g_{av} = \sum_{i \in I} \frac{d(i)}{p(i)}$,

irrespective of phase selection. This holds because a set of p consecutive frames contains $\frac{p}{p(i)}$ workloads for customer i, and therefore a total workload duration of $\sum_{i \in I} \frac{p}{p(i)} d(i)$. Dividing by p gives g_{av}.

Let $\{f(1), f(2), ..., f(|I|)\}$ be any assignment of phases for the customers, and define the associated x by (1-4). Let $g_l(x)$ and $g^h(x)$ then denote the minimum and maximum values, respectively, of the p righthand sides of (1-3c), i.e., the minimum and maximum frame workloads under assignment x. Note that (1-3) may be reformulated as:

(3-2) $\quad z = min_x g^h(x)$

because the pair $\{x, y = g^h(x)\}$ is feasible for (1-3).

Lemma 1: Bounds on z.

For any allocation x satisfying (1-3b,d),

(3-3) $\quad max[g_{av}; \sum_{i,p(i)=1} d(i) + max_{i,p(i)\geq 2} d(i)] \leq z \leq g^h(x) \leq g_{av} + g^h(x) - g_l(x)$

Proof: The lower bound $g_{av} \leq z$ states that the average frame workload $\leq$ the largest frame workload. A formal derivation consists of summing (1-3c) over k = 1,2, ... p and using (1-5) and (1-3b) to obtain:

$$py \geq p \sum_{i \in I} \frac{d(i)}{p(i)} \text{ or } y \geq g_{av}.$$

The other lower bound,

$$\sum_{i,p(i)=1} d(i) + d(k) \leq z \text{ where k = argmax } max_{i,p(i)\geq 2} d(i)$$

is obtained by noting that if customer k is served in some frame m, then z must be at least as large as the load of frame m, which is at least:

$$\sum_{i,p(i)=1} d(i) + d(k).$$

The bound $z \leq g^h(x)$ is from (3-2) while the rightmost inequality in (3-3) follows from the relationship $g_{av} - g_l(x) \geq 0$, i.e., the average frame workload is equal to or greater than the smallest frame workload. QED.

The bounds $g_{av} \leq z \leq g_{av} + g^h(x) - g_l(x)$ shows that any assignment x is ε-optimal, with $\varepsilon = g^h(x) - g_l(x)$. An additional objective in this type of periodic loading problems is to seek an assignment which attempts to minimize the difference $g^h(x) - g_l(x)$ between maximum and minimum frame workloads. In this case a frame corresponds to a worker, the fairness of workload assignment is perceived mainly by the size of ε whose minimization might become the main objective. In addition, the attempt to keep all p frames equally loaded can serve as a surrogate to minimizing the maximum load. We refer to heuristics which attempt to minimize ε as *balance-seekers*.

To avoid misunderstanding, we illustrate below that minimizing $g^h(x) - g_l(x)$ is **not** equivalent to minimizing $g^h(x)$.

Counter example, $I = \{1, 2, 3\}$

i	p(i)	d(i)
1	2	5
2	2	6
3	4	8

Complete enumeration shows that $z = min_x g^h(x) = 11$ is achieved at $\{f(1), f(2), f(3)\} = \{1, 1, 4\}$ with $g^h = 11, g_l = 0$ and imbalance $g^h - g_l = 11$. The minimum load imbalance

$$min_x[g^h(x) - g_l(x)] = 8$$

is lower, and achieved at $\{f(1), f(2), f(3)\} = \{1, 2, 1 \text{ or } 3\}$ with $g^h = 13, g_l = 5$ and therefore a larger value of g^h.

4. GREEDY HEURISTICS

Introduction

Since the PLP is NP-complete, exact solution techniques (branch-and bound) can be complex. This motivates the search for heuristic algorithms which produce solution of acceptable quality for practical purposes. This section describes a class of greedy heuristics while the next section supplies some additional heuristics.

Greedy Heuristics

The following class of greedy heuristics is especially easy to program.

Initialization: Select one of the following three ordering schemes: order of nondecreasing p(i), or of nonincreasing d(i), or of nondecreasing $\frac{p(i)}{d(i)}$.

Iterative Step: Proceed through the set I of customers in the sequence generated above. Select the phase for the current customer to minimize the increase in $g^h(x)$ (or $g^h(x) - g_l(x)$ for balance-seeking). Note: the g's are computed with $x_{ij} = 0$ for all j whenever customer i has not yet been assigned a phase. Break ties arbitrarily.

Greedy Heuristic for Power-Period PLP - (nondecreasing p(i))

The above heuristic simplifies if each period $p(i) \in \{1, r, r^2, ..., r^k = p\}$, if the customers are assigned to the most lightly-loaded frame. Such a scheme simultaneously minimizes the increases in g^h and in $g^h - g_l$ (hence is a balance-seeker), produces convenient schedules as in Example 1, and has the additional properties given by Lemma 2 below.

The heuristic is:

Initialization: Label all customers "unassigned". Set m = -1.

LOOP 1: Increase m by 1. Terminate if $m > k$.

LOOP 2: If all customers with period r^m are assigned, go to LOOP 1. Otherwise continue.

Let j^* denote the most-lightly-loaded frame among frames 1 through r^m. (Break ties for j^* arbitrarily.) Select any unassigned customer i^* with period r^m (break ties for i^* arbitrarily, e.g., largest d(i)) and assign i^* to phase j^*. Return to LOOP 2.

Denote by $D = max_{i \in I, p(i) \geq 2} d(i)$

Lemma 2: The preceeding algorithm has the properties:

(a) At any stage in the assignments, the difference in workload between the most-heavily-loaded frame and the most-lightly-loaded frame is at most D.

(b) If x is the allocation produced by this heuristic, with $g^h(x)$ again denoting the load on the most-heavily-loaded frame, then

(4-1) $$g_{av} \leq z \leq g^h(x) \leq g_{av} + D,$$

i.e., the algorithm is guaranteed to come within D of the optimal solution.

Proof: Let x denote a partial solution, with $x_{ij} = 0$ for all j if customer i has not yet been assigned. We will show inductively that:

(4-2) $$g^h(x) - g_l(x) \leq D$$

which implies (a). Then (b) follows from Lemma 1 and (4-2). We ignore customers with p(i) = 1.

The very first assignment, say of customer k, satisfies (4-2) because $g^h(x) = d(k) \leq D$ while $g_l(x)$ is 0 since $p(k) \geq 2$.

For the inductive proof of (4-2), let x^{old} and x^{new} denote the allocation in LOOP 2 before and after customer i^* is assigned to phase j^*. By assumption, (4-2) holds for x^{old}.

After assigning customer i^* to phase/frame j^*, we find

$$g_l(x^{new}) > g_l(x^{old})$$

(since the load has increased) while

$$g^h(x^{new}) = max[g_l(x^{old}) + d(i^*), g^h(x^{old})]$$

where the two terms on the right are the cases where the new load $g_l(x^{old}) + d(i^*)$ on frame j^* is/isn't the new maximum load. Subtracting,

$$g^h(x^{new}) - g_l(x^{new}) \leq max[d(i^*), g^h(x^{old}) - g_l(x^{old})] \leq max[D, D] = D$$

using the definition of D and (4-2) to get the last inequality. QED.

Equation (4-1) shows that this algorithm produces a D-optimal solution, hence yields good solutions if D is small compared with g_{av}. This is understandable since one can do nearly perfect load-balancing if job durations are very short.

Corollary 1: Given identical message durations, i.e., d(i) = d for all i, the Greedy Heuristic for the power-period PLP is optimal for any way of breaking ties.

Proof:

Under each allocation, the load on each frame is an exact multiple of d. Let x denote the allocation produced by the greedy heuristic. There exist integers s, t, u such that $z = sd, g^h(x) = td, g_l(x) = ud$. Obviously $t \geq u$, and Lemma 2(a) implies $t - u \leq 1$. Hence either $t = u$ (Case 1) or $t = u + 1$ (Case 2).

Case 1: (3-3) implies $g_{av} \leq z \leq g^h(x) \leq g_{av} + 0$

Hence $z = g^h(x)$ and x is optimal.

Case 2: Since g_{av} is independent of the allocation, the properties $g_l(x) = ud, g^h(x) = (u + 1)d$ imply $h_{av} > ud$. Then (4-1) implies $ud < g_{av} \leq z = sd \leq g^h(x) = (u + 1)d$. Hence $s = u + 1, z = g^h(x)$, and x is optimal. QED

A very similar algorithm with the ordering scheme of nonincreasing d(i) is also constructed for the power-period PLP. We denote these two heuristics as Greedy p(i) and Greedy d(i) respectively.

Note: The above Greedy Heuristic (together with four other heuristic algorithms) was introduced in an earlier version of this paper [14]. That algorithm was used later

in the context of the economic lot sizing problem (ELSP) in [3].

5 OTHER HEURISTICS

This section contains other heuristics, more complex than those in the previous section: merging, unbalanced greedy, and linear programming with roundoff. Of course, hybrids of these heuristics are possible as well. For example, use the merge algorithm to assign phases to a subset of the customers (e.g., to the n largest ones), and next use a greedy algorithm to handle the remaining customers. Because these heuristics are fast and easily programmed, it is practical to employ many of them and retain the best solution produced. For example, the value of "n" may be varied in the merge hybrid described two sentences ago. Some of this hybrid flavor is introduced in the unbalanced greedy algorithm.

Merge Heuristic for Power-Period PLP

Consider the following power-period example with r = 2, $|I| = 6$ and p = 32, with customers listed in order of decreasing period.

i	p(i)	d(i)
A	32	12
B	32	10
C	8	14
D	8	9
E	4	20
F	2	16

The two customers A and B are to be *paired off* into complementary frames (e.g., f(A) = 3 and f(B) = 19). This amounts to restricting the original problem (1-3) with the additional constraint $|f(A) - f(B)| = 16$, and essentially replaces A and B by a single "composite" message G with period p(G) = 16 and duration d(G) = max[10, 12] = 12. Such merging may be a convenient way of load-balancing, by pairing off the longer customers. Merging can be continued, with C then paired off with D (or G). Merging reduces the size of the list until the list reduces to a few composite members which can be assigned phases by inspection. Disaggregation then yields phases for the original members.

The *general merge heuristic* for periods which are powers of r is as follows:

1. initialization: Start with a list of r or more customers (e.g. subset of the longest customers). Label all customers on list as "not composite".

2. sort step: Sort the list in order of nonincreasing period p(i), with ties broken in order of nonincreasing d(i).

3. If the list has at most r - 1 members with $p(i) \geq r$, set f(i) = 1 if p(i) = 1, and allocate the customers with $p(i) \geq r$ over the r frames so that imbalance is les or equal to D. Go to disaggregation step 5. (See remark 1 below.)

4. merge step:

(a) Replace the first r entries on the list, say customers $\alpha_1, \alpha_2, ..., \alpha_r$ by a new customer γ with

$$p(\gamma) = \frac{1}{r} min[p(\alpha_1), p(\alpha_2), ..., p(\alpha_r)] \quad \text{and } d(\gamma) = max[d(\alpha_1), d(\alpha_2), ..., d(\alpha_r)]$$

(b) Label γ as a "composite" of $\alpha_1, \alpha_2, ..., \alpha_r$.

(c) Return to sort step 2. (See remark 2 below,)

5. disaggregation step:

(a) Enter here with a list of customers, some composite and some non-composite, each of which is assigned a phase f(.).

(b) If every customer on the list is non-composite, terminate. Otherwise continue.

(c) Select any composite customer γ from the list. If γ is a composite of customers $\alpha_1, \alpha_2, ..., \alpha_r$, then remove γ from list and add $\{\alpha_1, \alpha_2, ..., \alpha_r\}$ to the list with their original d's and with $f(\alpha_j) = f(\gamma) + (j-1)p(\gamma)$, j = 1,2, ..., r. Return to disaggregation step 5a.

Remark 1. An alternative criteria for branching is whenever the list is short enough that good phase assignments f(.) are possible. For example, if every customer on the list has p(i) = 1 or 2, set f(i) = 1 whenever p(i) = 1 and find the best phases for the customers with p(i) = 2 via the set partitioning algorithm (cf. Section 2).

Remark 2. The sort step is merely to delete $\alpha_1, \alpha_2, ..., \alpha_r$ from the list, and insert γ in the appropriate place.

Linear Programming Relaxation

By relaxing (1-3d) to $x_{ij} \geq 0$, (1-3) reduces to a linear program (LP). Since the LP relaxation allows *splitting* of a customer among several frames, perfect *load-balancing* can be achieved: the optimal value of the LP is $z^{LP} = g_{av}$, achieved when

$$(5\text{-}1) \qquad x_{ij} = \frac{1}{p(i)} \qquad 1 \leq j \leq p(i), 1 \leq i \leq |I|$$

namely, when each customer is equally divided among all possible phases. Every frame has workload g_{av}: under (5-1), each right side of (1-3c) is g_{av} due to (1-6).

Although neither z^{LP} nor the allocation (5-1) are of interest, the optimal *basic* feasible solution to the LP is of interest. Unlike (5-1), many of its elements are 0

or 1 , and intelligent rounding of remaining entries gives a reasonable solution. This approach has been used successfully, for example, in production planning [10] and in a version of the generalized assignment problem [2] based on result from Lasdon [9, p.173]. In addition, the LP relaxation is convenient for an exact branch and bound procedure where the values of certain x_{ij} are fixed.

In our own numerical experimentation, the rounded LP solution was roughly comparable in quality to that produced by the greedy heuristics. The principal difficulty encountered was that the LP tended to split the longest, rather than shortest customers and no simple rounding rule was apparent.

The rounding rule adopted was to (i) remove any split customers from the allocation, while keeping any non-split allocations; (ii) list the split customers in order of nonincreasing d(i) with ties broken in order on nondecreasing p(i); (iii) reallocate the split customers in this sequence using the greedy algorithm for minimum increase in g^h.

Lemma 3: The LP solution together with the rounding rule guarantee to come within D of the optimal solution and imbalance no greater than D.

Proof:

Case 1: All the customers are split in the LP solution. In this case the greedy heuristic guarantees the bounds on the solution.

Case 2: A proper subset of customers are split in the LP solution. Here we examine two subcases: (i) The initial non-split allocation, say x_0, has an imbalance less or equal to D, in which case the greedy heuristic insures the claim, (ii) The initial non-split allocation, x_0, has an imbalance greater than D. In this case it is obvious that each subsequent greedy allocation reduces the imbalance until it is at or below D. QED.

6. UNBALANCED GREEDY ALGORITHM FOR THE POWER-PERIOD PLP

The greedy algorithm for the power-period PLP given at the end of Section 4 has the feature that for *each* period, it seeks to balance the workload across frames: it first does balancing with p(i) = 2, then continues its attempt to balance with p(i) = 4, etc. However, load-balancing at every period is not necessary and in fact may be undesirable. For example, if there is one very large customer with (say) p(i) = 8, a better strategy is to set aside some of the space in frames 8, 16, 24, 32, ... to accommodate this large customer. This implies imbalance at the smaller periods: when p(i) = 2, phase 1 should receive heavier usage than phase 2, and when p(i) = 4, phases 1,2,3 should receive heavier usage than phase 4, because of the anticipated need for frame 8. In essence, it implies that allocation of customers to frames should

reflect in each step not only the immediate optimizing opportunities but also the subsequent demand for frames.

This is the central theme in this section and is apparent in this radically different algorithmic approach for solving the power-period PLP. The algorithmic procedure, referred to as the Unbalanced Greedy Algorithm, is an attempt to construct a more "intelligent" way of assigning customers to frames in the context of a power-period PLP. Some of the ideas found in the field of Artificial Intelligence [13] are adopted here in this algorithmic development.

The Unbalanced Greedy Algorithm

The algorithm consists of a data preprocessing stage and an assignment stage which is also data sensitive.

Data Preprocessing Stage

In this stage the messages are sorted, partitioned, and merged to form new messages. The date is examined in relation to average message duration and maximal (minimal) message duration.

Denote the average message duration by $d_{av} = \sum_{i \in I} \frac{d(i)}{|I|}$, $d_{max} = max_{i \in I} d(i)$, $d_{min} = min_{i \in I} d(i)$.

(Note not to confuse d_{av} with g_{av}.)

If the duration of all the messages is roughly the same, one would expect the heuristic algorithms described in Sections 4 and 5 to produce a 'satisfactory' solution. Since the above algorithms (except for the LP based heuristic) are 'fast' in terms of CPU time, we select in this case the best solution produced by all four heuristics. The decision rule adopted for the Unbalanced Greedy Algorithm is that if the longest message is not much longer than the average message ($d_{max} \leq 1.3 d_{av}$), then *Go To* the four heuristic algorithms and select the best solution.

In case some of the messages are long ($d_{max} > 1.3 d_{av}$) then the messages are partitioned according to their periodicity and relative duration. First, two message categories are constructed by message duration:

$$I_1 = \{i | \frac{d(i) - d_{min}}{d_{max} - d_{min}} \geq 0.7\} \text{ and } I_2 = I \setminus I_1$$

Denote by $I^i = \{j | r^j = p(j)\}$, $i = 1, 2, ..., k (r^k = p)$, then a more refined partition is obtained as follows:

$$I_1^i = I^i \cap I_1 \text{ and } I_2^i = I^i \cap I_2 \quad \text{i = 1,2, ..., k.}$$

Sort the set I_1^i by nonincreasing duration. Given this initial message partition, the preprocessing algorithm checks for pairs of messages in I_1^i (starting with i = k) and merges each such pair into a single message in I_1^{i-1} and updates the message

partition. In case the set I_1^i contains only one message, the set I_2^i is searched for an 'appropriate' combination of (short) messages to pair it (merge it) with the message (say j^*) in I_1^i. To find such a message combination a knapsack problem is formulated and solved:

$$\text{Max } z = \sum_{j \in I_2^i} y_j$$

Subject to

$$0.8d(j^*) \leq \sum_{j \in I_2^i} d(j) y_j \leq 1.1 d(j^*)$$

$$y_j = 0 \text{ or } 1$$

If a solution to the knapsack problem exists, then the combination of the short messages (the ones for which $y_j = 1$ in the solution) is merged with j^* to become a new message in I_1^{i+1}.

At the conclusion of the preprocessing stage a 'new' message might be composed of a number of the original messages.

The Message Assignment Stage

In the unbalanced frame assignment heuristic we anticipate the assignment of 'free' long messages when selecting a frame assignment for the message at hand. This heuristic algorithm first assigns the messages in I in a nondecreasing order of p(i) and then repeats the process with medium duration messages before assigning the short messages. It follows the following steps:

1. initialization: List the messages in I in order of nondecreasing p(i) and break ties by nonincreasing d(i).

2. frame ranking: Given a partial assignment list X, denote by $d(i_1^*) = max\{d(i) | i \in I \setminus X\}, d(i_2^*) = max\{d(i) | i \neq i_1^*, i \in I \setminus X\}$, (i.e., the two longest messages yet not assigned). Denote by $g^h(X)$ the maximal frame load given the partial assignment X. The number of frames need to be examined is given by $\max\{p(i_1^*), p(i_2^*), p(j^*)\}$, j^* being the current message on the list.

Denote by $\triangle_1$ the increase in $g^h(X)$ if selecting the optimal frame for the message j^*. Similarly, $\triangle_2$ if selecting frames for both j^* and i_1^*, $\triangle_3$ if simultaneously selecting frames for j^*, i_1^*, and i_2^*. Given the partial assignment X, the values $\triangle_i, (i = 1,2,3)$, are calculated by refined enumeration.

Obviously, $\triangle_3 \geq \triangle_2 \geq \triangle_1$.

If we choose for message j^* the frame which corresponds to $\triangle_3$ it means that we attempt to anticipate the assignment of messages i_1^* and i_2^*. Similarly, if we select the frame corresponding to $\triangle_2$. A frame is assigned a rank 1 if for j^* it corresponds to $\triangle_1$, rank 2 if it corresponds to $\triangle_2$, and rank 2.5 if it corresponds to $\triangle_3$. (All other frames have a rank 0.) Since the same frame might correspond to a number of $\triangle$'s,

the rank is additive. The purpose of a such rank is to 'measure' the attractiveness of a frame assignment for message j^*.

3. message assignment: For the message j^* select the frame with the highest rank and *continue*.

7. THE POST PROCESSING ALGORITHM

So far we have described five heuristic procedures which generate a solution for the power-period PLP. All five procedures construct a feasible assignnment of messages to frames starting initially with empty frames. A post processing algorithm requires, in principle, a feasible solution as an input and then attempts to improve it without violating the feasibility of the solution. For a number of 'hard' combinatorial problems, the solution approach based on a 'main' heuristic algorithm followed by a post optimization process proved highly successful (for example see [2]). This is true also in the case of the PLP. In this section we describe the post optimization algorithm adopted for the PLP.

1. initialization: Fix integer parameters a and b. Start with a feasible solution (i.e., all messages are assigned to frames). Select the (first) frame with load equal to g^h.

2. candidate step: Sort the messages in the frame in nondecreasing order of p(i) breaking ties in nondecreasing order of d(i). The first message on the list is say j^*.

3. move step: We might examine up to $f(j^*) + p(j^*) - 1$ potential frames for moving message j^*. (Message j^* has a period $p(j^*)$ so only $p(j^*) - 1$ frames are distinct for it.)

Move j^* to the next frame. If the number of frames with load at or above g^h exceeds a then move j^* to the next potential frame. If all potential frames were examined, repeat the process with the next message on the list. If no frame has a load of g^h or above assign j^* to the present frame and repeat step 1.

4. If the number of frames with load greater or equal to g^h is between 1 and a and only one message has been moved then select the first frame at or above g^h and repeat step b. This is likely to result in a message other than j^* being moved. If the number of messages moved is already 2 then drop j^* (0 messages moved) and repeat the process with the next message on the list.

This local improvement search continues until the list in Step 2 is exhausted.

There are two parameters in Step 4 of this procedure: a and b. These parameters have a considerable impact on the quality and CPU time required. In all the computational results reported in this paper a = 6 and b = 2.

8. COMPUTATIONAL ILLUSTRATION

Analysis of the avionics system on the F-16 fighter aircraft illustrates the above ideas. Nine computers, with 33 types of messages (customers), are interconnected via a 1 megabit-per-second communications bus employing MIL-STD-1553 TDM protocol [15, Table I]. Frame size was T = 20 milliseconds (50 frames per second) and message periods and lengths are given in Table 1. Note this is a power-period problem with r = 2, p = 32, and $|I| = 33$.

TABLE 1

Data for F-16 Example

Message Type i	Period p(i)	Message Length d(i) (in words)	Message Type i	Period p(i)	Message Length d(i) (in words)
1-16	1	225	25	4	6
17	2	12	26	8	30
18	2	13	27	8	31
19	2	6	28	8	33
20	2	9	29	8	5
21	2	13	30	32	27
22	4	9	31	32	31
23	4	10	32	32	6
24	4	12	33	32	3

Each word (20 bits of data) requires 20 microseconds to transmit. For simplicity of exposition, overhead (intermessage gap) has been ignored and frame loads are measured in words. Message types 1 through 16 each had periods of one, and have been lumped together without loss of generality since $x_{i1} = 1$ if $p(i) = 1$. Message types 18 and 21 have identical characteristics but this was not exploited.

The average load per frame is obtained from (3-1) as

$$g_{av} = 275.22 \text{ words per frame}$$

which corresponds to an average frame utilization of 27.522% (since each frame holds up to 1,000 words). Of this, 225 words per frame (utilization of 22.5%) is due to the messages with period one.

Table 2 contains the results of six runs attempting to solve (1-3) for the above problem. The best heuristic, run 4, leads to the bounds:

$$276 \leq z \leq 278$$

which fix z within 1% (cf. (3-3); the lower bound 275.22 = $g_{av} \leq z$ has been rounded upwards since z is an integer. (If the load of 225 words per frame due period-1 messages is discarded, the bounds $51 \leq z \leq 53$ fix z to within 4%.) Note that run 6 has the smallest value of $g^h - g_l$. A detailed discussion of each run now follows. For each run we report the results before and after post processing denoted by bp and ap respectively.

TABLE 2

Computational results for F-16 Example

Message i	Run 1 f(i)		Run 2 f(i)		Run 3 f(i)		Run 4 f(i)		Run 5 f(i)		Run 6 f(i)	
	bp	ap	bp	ap	bp	ap	bp	ap	bp	ap	bp	ap
1-16	1	1	1	1	1	1	1	1	1	1	1	1
17	1	1	1	1	2	1	1	1	1	1	1	1
18	1	1	2	2	2	1	1	1	1	1	1	1
19	2	2	1	1	2	2	2	1	1	1	1	1
20	1	1	1	1	1	1	1	1	2	2	1	1
21	1	1	2	2	1	1	1	1	1	1	1	1
22	4	2	1	2	2	1	4	4	1	1	2	2
23	4	4	4	2	2	1	4	4	2	2	2	2
24	1	1	2	2	4	3	2	2	4	4	4	4
25	2	2	3	3	4	3	2	2	3	3	4	4
26	4	4	7	5	4	4	2	2	4	4	2	2
27	8	8	1	1	3	6	6	6	2	2	4	4
28	6	6	4	4	1	2	8	8	6	6	6	6
29	2	2	3	3	7	4	1	2	3	3	8	8
30	10	10	13	8	5	8	12	12	8	8	8	8
31	2	2	16	16	2	16	28	28	24	24	16	16
32	28	28	8	8	8	8	20	20	16	16	24	24
33	12	12	5	3	13	2	4	4	2	2	32	32
g^h	286	284	294	284	294	279	283	278	280	280	279	279
g_l	252	251	261	251	259	243	253	247	246	246	251	251
$g^h - g_l$	34	33	33	33	35	36	30	31	34	34	28	28

Run 1 used the LP relaxation rounded as described in Section 5. Note that the quality of this solution g^h is only slightly better than that of the simple greedy heuristics, run 2 and run 3, namely 10% too high. The basic feasible solution to the LP had 28 non-split customers plus 5 split customers. The latter were split into 2-7 phases, an average 4.8 phases per customer, leading to a total of 24 fractional x_{ij}. A different reassignment of split messages to frames was implemented in this run. It examined only the subsequent increase in the maximally loaded frame but not the minimally loaded frame. Thus, the imbalance can be greater than d_{max}, which is the case here. $g^h - g_l = 34$.

Run 2 used the greedy heuristic for the power-period PLP given at the end of Section 4: messages $17 \leq i \leq 33$ are listed in order of nondecreasing p(i) with ties broken in order of nonincreasing d(i) and are assigned to the lightest-load frame. Note that $g^h - g_l = 33 \leq 33$, in agreement with (4-2).

Run 3 used the greedy heuristic at the beginning of Section 4 to minimize increases in g^h, with messages $17 \leq i \leq 33$ listed in order of nonincreasing d(i) and ties broken in order of nondecreasing p(i). Note (4-2) is violated when this listing is employed.

Run 4 first applied the merge algorithm of Section 5 to the messages $17 \leq i \leq 33$. This algorithm with a 'little' help from the post processor produced the best solution of $g^h = 278$.

Run 5 is the unbalanced greedy heuristic described in Section 6. It produced a very good starting solution of $g^h = 280$, but unfortunately the post processing algorithm did not find an improvement.

Run 6 illustrates an ad hoc unbalanced greedy algorithm, in which frames 8, 16, 24, and 32 were pre-assigned to (long) messages i = 30, 31, 32, 33, with p(i) = 32 (i.e., these 4 messages were merged into a composite message with period 8). Note the allocation of messages with p(i) = 2 was unbalanced, all going to odd-numbered frames. Similarly, the allocation of messages with p(i) = 4 was essentially balanced among the even-numbered frames, with no use made of the odd-numbered frames. Similarly, the allocation of messages with p(i) = 8 was essentially balanced among the even-numbered frames. This ad hoc solution illustrates the value of a look-ahead strategy. The post processing algorithm did not find an improvement in this case either.

9. COMPUTATIONAL EXPERIMENTS

The illustration of algorithmic performance on the F-16 avionics system does not constitute a large computational experiment. Though the problem is taken from a real evironment, it does not provide sufficient empirical evidence about the performance of the heuristics described in this paper. In order to test the performance of the five algorithms, we conducted the following computational experiment: Three sets of 25 problems each with 50 messages were solved using the above five heuristcs. The

three sets differ in the message duration parameters. In each of the problems the periodicity was generated randomly according to a uniform distribution among the numbers 2,4,8,16, and 32. In addition, five problems were generated with maximal periodicity of 16 and five more problems with maximal periodicity of 64. The purpose was to test the sensitivity of the algorithmic performances with respect to the maximal period. Overall 85 different problem instances were generated and each solved with the five algorithms. Each solution obtained by a 'main' algorithm was subjected to the post processing procedure. We report for each problem and algorithm the results obtained before and after the post processing.

For the first set of the 25 problems, the duration for each message was obtained by rounding up to an integer the number randomly generated from a uniform distribution between 18 and 24 (i.e., a narrow band of durations). For the second set of the 25 problems, the parameters for the uniform distribution were changed to 2 and 40 respectively. In the third set of the 25 problems the message durations were generated from three different uniform distributions: 22 of the messages had durations generated with parameters 2 and 10, 6 with parameters 11 and 31, and 22 with parameters 32 and 40. In Tables 3, 4, and 5 we present the solution values before and after post processing on all 25 problems for each message duration parameter set respectively. The values in the row denoted by L.B. are the (reference) lower bound values for each problem. The lower bounds are equal to $\max\{max_{1\leq i\leq 25} d(i), \lceil \sum_{i=1}^{50} \frac{d(i)}{p(i)} \rceil\}$. The respective algorithms are denoted as follows: L.P. is the linear programming based heuristic, G. p(i) is the heuristic which sorts the messages in a nondecreasing order of p(i), G. d(i) is the heuristic which sorts the messages in a nondecreasing order of d(i), Me. stands for the merge heuristic, and Ub. G. denotes the unbalanced greedy heuristic.

([18-24]) The Algorithm		Problem Number 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	No of I''
L.B.		162	220	168	193	144	171	190	171	208	182	200	250	176	216	170	166	171	193	205	193	217	182	201	249	184	
L. P.	bp	171	230	179	210	161	185	207	188	228	193	210	267	187	225	186	171	189	209	212	203	232	194	215	261	205	3
	ap	167	226	171	197	150	176	194	176	213	189	205	262	184	222	174	168	178	204	209	198	224	188	210	260	188	8
G. p(i)	bp	171	231	180	206	156	186	205	182	216	194	210	267	187	229	181	170	188	206	213	202	228	192	211	255	192	8
	ap	166	227	174	201	151	184	193	175	212	186	206	259	184	220	177	168	182	198	209	198	225	187	208	253	187	11
G. d(i)	bp	169	228	188	210	161	189	204	186	212	189	208	265	190	231	184	168	187	209	208	208	230	188	211	253	203	10
	ap	168	227	171	206	151	183	197	184	212	187	208	263	181	228	179	168	183	196	208	198	224	186	208	252	187	6
M e.	bp	169	228	171	210	162	187	205	189	212	190	209	265	191	231	185	167	187	209	208	209	230	187	213	252	188	11
	ap	168	226	170	203	150	180	195	190	212	188	207	258	185	223	182	167	185	200	207	201	228	187	209	252	188	7
Ub. G.	bp	169	228	171	206	156	185	204	182	212	189	208	265	187	225	181	167	187	206	208	202	228	187	211	252	188	25
	ap	168	226	170	201	151	176	197	175	212	187	208	258	184	222	177	167	185	198	207	198	225	187	208	252	188	15

([2-40]) The Algorithm		Problem Number 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	No of I''
L.B.		208	162	217	218	152	240	143	254	224	235	207	219	163	180	164	228	190	157	241	298	222	261	237	188	226	
L. P.	bp	226	179	236	233	175	251	163	271	233	250	228	235	189	199	181	251	208	178	271	310	237	273	255	220	243	0
	ap	211	166	220	227	155	249	147	262	227	242	212	223	167	186	169	237	197	161	244	304	226	269	242	194	231	5
G. p(i)	bp	229	186	230	240	176	270	159	280	237	256	229	233	182	204	176	242	211	178	263	321	241	278	255	207	246	0
	ap	212	165	225	228	155	246	147	259	228	239	212	229	169	189	171	232	194	163	247	304	226	267	243	197	234	2
G. d(i)	bp	211	169	226	226	159	252	148	260	235	241	217	223	170	184	170	233	199	160	249	302	228	264	245	194	229	16
	ap	210	166	221	226	156	245	148	259	229	240	212	221	167	184	169	231	193	159	247	302	224	264	242	192	229	13
M e.	bp	216	168	226	230	158	249	148	261	229	247	214	226	173	186	174	235	200	162	249	308	226	269	244	195	231	9
	ap	211	164	221	221	155	246	145	258	227	240	210	223	167	184	167	231	193	160	246	300	225	263	241	192	229	14
Ub. G.	bp	217	176	222	230	160	258	154	267	231	248	220	231	173	190	174	239	198	168	258	309	236	271	248	204	240	2
	ap	212	172	220	225	161	246	146	262	228	239	212	223	169	186	171	231	195	159	245	302	230	265	243	191	229	7

(3 U Dist.) The Algorithm		Problem Number																									No. of 1st
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	
L.B.		235	215	223	223	191	192	186	151	168	175	184	159	200	202	233	163	174	226	198	221	178	237	192	255	278	
L.	bp	248	242	235	241	211	209	200	171	178	193	201	189	223	224	261	180	194	240	230	245	204	255	216	283	312	0
P.	ap	236	225	235	228	193	201	191	158	171	182	186	163	201	205	250	166	177	233	202	226	183	243	208	261	285	6
G.	bp	253	228	248	245	211	216	209	163	186	193	210	173	217	222	256	178	197	246	217	240	195	265	214	281	295	0
p(i)	ap	237	222	232	228	198	200	193	155	171	180	185	161	207	209	239	169	179	239	203	227	190	246	198	265	286	3
G.	bp	239	218	238	231	196	196	190	158	176	184	196	162	210	204	242	171	177	233	199	225	186	247	197	261	283	14
d(i)	ap	236	218	231	231	195	195	189	158	175	181	196	161	210	204	240	170	175	233	199	225	186	243	197	260	281	10
M	bp	241	223	232	233	199	207	197	164	176	187	192	168	208	207	238	175	185	236	207	233	184	241	201	274	289	5
e.	ap	236	220	226	223	194	195	187	154	171	179	189	165	205	204	236	166	176	234	202	225	182	238	195	261	281	15
Ub.	bp	245	228	232	226	202	195	189	156	179	183	202	167	204	208	248	180	179	235	202	235	184	241	211	259	290	9
G.	ap	242	221	230	225	198	196	188	153	177	179	187	162	203	205	237	180	176	233	200	229	183	240	196	258	288	5

From the above tables it is clear that the post processing algorithm has a significant impact on the solution generated by the 'main' algorithms. For over 90% of the problems the post processing algorithm was able to reduce the g^h value. In a few cases this reduction was by as much as 10%. The last column in Tables 3,4, and 5 presents the number of times (out of 25) each algorithm produced the best solution. Since there is no clear dominance of one algorithm over another, and all the algorithms give reasonable results (in most cases, well under 5% above the lower bound) the final recommendations as to which algorithmic approach is appropriate in this case has to take into account the CPU time involved. In Table 6 we provide the average CPU time in seconds for each of the algorithms. In addition, the minimum and the maximum CPU times are also given. The times in Table 6 correspond to the three sets of 25 problems each (i.e., 75 problems). All the computations were conducted on a CYBER 173 using a PASCAL 6000 compiler.

Table 6

The summery of the CPU times (in sec.) for the 75 problems

The * over the column in Ub.G. indicates that those computational results represent only 50 problems from the Tables 4 and 5.

	CPU Time (sec)				
	L.P.	G. p(i)	G. d(i)	Me.	Ub. G.*
Ave.	46.39	0.32	0.19	1.33	154.2
Min.	32.70	0.27	0.17	1.07	21.05
Max.	64.39	0.38	0.25	1.66	944.10
	Post Processing CPU Time (sec)				
Ave.	14.00	16.65	12.08	9.06	10.42
Min.	1.32	3.80	0.51	0.22	0.19
Max.	57.33	43.28	60.17	40.37	37.97

Since the Ub. G. heuristic takes the best solution obtained by the other four algorithms in case of data parameters for the problems in Table 3, we do not mix those CPU times with the other two problem sets. We do not report the computational results for the two five problem sets with a maximal periodicity of 16 and 64 respectively. The objective in that experiment was to examine whether any of the algorithms described here demonstrates superiority over the others when decreasing or increasing the maximal number of frames. This was not the case. The algorithms did not exhibit any different tendencies on the 16 or 64 frame problems than on the 32 frame problems. The one important point that should be made is that for the 64 frame problems the Unbalanced Greedy Algorithm required up to 2842 CPU seconds to reach a solution.

10. CONCLUSIONS

We presented in this paper formulation and heuristic algorithms for the Periodic Loading Problem emphasising the Power-Period PLP. The five algorithms described and analyzed here, differ considerably from each other (except the two 'sorting' algorithms). The algorithm of special interest is the Unbalanced Greedy. In this algorithm the solution method is both data sensitive and has a limited look ahead property. Both of these features in an algorithm present a novel approach for solving this type of hard combinatorial problems.

All five of the algorithms were tested extensively on artificially constructed problems. The conclusions that we draw from the results of these computational experiments are as follows: The fast 'sorting' algorithms (the G. p(i) and G. d(i)) performed very well in comparison with the more 'sophisticated' algorithms (the L.P. and Ub. G.). The Merge Algorithm also compares well as far as its results and the CPU time required. (The L.P. and Ub.G. cannot be dismissed out-of-hand. since in atleast on a small fraction of the cases they outperformed the faster heuristics. Such superior results, however, were typically only a few percent better than the faster heuristics.) It is interesting that the fastest algorithm, the G. d(i), has the over all highest first place finishes among the algorithms. The post processing algorithm, though much slower than the fast heuristics themselves, proved to be a very significant factor in obtaining quality solutions for all the five algorithms. We recommend that such an algorithm be adopted as a final algorithmic stage with any 'main' solution procedure, since the extra few seconds of CPU time are acceptable in most cases. Since the heuristics G. p(i), G d(i), and Merge, are fast procedures, one alternative could be to apply them in parallel to a problem instance together with their post processing , and then selecting the best result.

Acknowledgements: The authors thank the referees for helpful suggestions. This research was initiated when the first author was a memeber of the Department of Industrial Engineering and Management of The Ben Gurion University of the Negev, which he thanks for many courtesies extended.

REFERENCES

[1] Baker, K.R., *Introduction to Sequencing and Scheduling*, John Wiley, New York (1974).

[2] Dror, M., "The Inventory Routing Problem", Working Paper MS/S - 83-028, College of Business and Management, University of Maryland at College Park (1983).

[3] Dobson, G., "The ELSP: A Resolution of Feasibility", Working Paper QM8504, William E. Simon Graduate School of Business Administration, University of Rochester, Rochester, N.Y. (1985).

[4] Elmaghraby, S.E., "The Economic Lot Scheduling Problem (ELSP): Review and Extension", **Management Science**, 24, 587-598 (1978).

[5] Garey, M.R. and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Francisco (1979).

[6] Gaudioso, M., Paletta, G., and Sanna, S., "Management of Periodic Demands in Distribution", **European Journal of Operational Research**, 20, 234-238 (1985).

[7] Gonzales, Jr., M.J., and Sch, J.W., "Periodic Job Scheduling in a Distributed Processor System", **IEEE Transactions on Aerospace and Electronic Systems**, A6S-12, 5, 530-535 (Sept. 1976).

[8] Hsu, W.L., "On the General Feasibility Test of Scheduling Lot Sizes for Several Products on One Machine", **Management Science**, 29, 93-105 (1983).

[9] Lasdon, L.S., *Optimization for Large Systems*, MacMillan, New York (1970).

[10] Manne, A.S., "Programming of Economic Lot Sizes", **Management Science**, 4, 115-135 (1958).

[11] Maxwell, W.L., and Singh, H., "The Effect of Restricting Cycle Times in Economic Lot Scheduling Problem", **IIE Transactions**, 15, 235-241 (1983).

[12] Park, K.S., and Yun, D.K., "Optimal Scheduling of Periodic Activities", **Operations Research**, 33, 690-693 (1985).

[13] Pearl, J., *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley Inc. (1984).

[14] Schweitzer P.J., and Dror, M., "The Periodic Loading Problem: Formulation and Heuristics", Working Paper QM8432, William E. Simon Graduate School of Business Administration, University of Rochester, Rochester, N.Y. (1984).

[15] Sundstrom, D.E., Anderson, W.B., Jr., Alford, S.A., General Dynamics, Forth Worth Division, "F-16 Multiplex: A System Perspective", Presented at 2nd AFSC Multiplex Data Bus Conference, 10-12 October 1978.